비기능적 요구사항을 강제하는 컴포넌트 개발 방법[1]

윤 석진, 신 규상
한국전자통신연구원, S/W 컨텐츠 융합 부문

Software Component Development Method Using Non-functional Requirements

Yoon, Seok Jin, Shin Gyu Sang
Electronics Telecommunication Research Institute
E-mail : sjyoon@etri.re.kr, gsshin@etri.re.kr

요 약

종래의 방법론은 분석대상 시스템의 기능적 요구사항을 추출하는데 초점이 맞춰져 있어서 상대적으로 시간 제한과 같은 개념은 다루어지고 있지 않다. 시간 제한과 같은 개념이 개발 시에 제대로 적용되지 않거나 간과하게 됨으로써 개발된 시스템의 성능은 예측 불가능하게 되며 시스템을 구현완료하고 테스팅 단계에서 시스템이 잘 못 개발되었다는 것을 확인하게 된다. 본 연구는 일반 업무용 소프트웨어 시스템의 개발에 있어서 비 기능적 요구사항으로 다루어지는 성능 관련 항목을 시스템 개발과정에 강제 시킴으로써 예측 가능한 소프트웨어 시스템을 구성할 수 있게 한다. 또한 하드웨어 중립적인 시간 개념을 사용함으로써 소프트웨어가 작동하는 하드웨어의 성능에 맞는 적절한 소프트웨어를 개발해 낼 수 있도록 한다.

1. 서론

일반적으로 업무용 시스템을 분석 개발하는데 있어서 모형화 방법을 사용한다. 이러한 모형화를 하는 방법으로서 객체지향 시스템 개발 방법론인 UML을 사용하는 방법론이 있으며, 이를 실시간이 요구되는 분야에 사용하기 위해 실시간 객체지향 방법론들이 있다. 컴포넌트는 컴퓨팅 시스템을 용이하게 구축하기 위해서 사용하는 소프트웨어 모듈로서 각각 독립적으로 사용될 수 있는 단위이다.

종래의 방법론은 분석대상 시스템의 기능적 요구사항을 추출하는데 초점이 맞춰져 있어서 상대적으로 시간 제한과 같은 개념은 다루어지고 있지 않다. 시간 제한과 같은 개념이 개발 시에 제대로 적용되지 않거나 간과하게 됨으로써 개발된 시스템의 성능은 예측 불가능하게 되며 시스템을 구현완료하고 통합 테스팅 단계에서 시스템이 잘못 개발되었다는 것을 확인하게 된다. 기존의 실시간 모형화를 지원하는 Rational Rose RT 와 같은 제품은 요구분석이나 설계 시보다는 구현단계에서 테

[1] 본 연구는 지식경제부 및 정보통신연구진흥원의 IT신성장동력핵심 기술개발사업의 일환으로 수행하였음. [2007-S032-02, 다중 플랫폼 지원 모바일 응용 S/W 개발환경 기술 개발]

스팅을 통해서 수행시간을 확인하는 방식으로 지원한다.

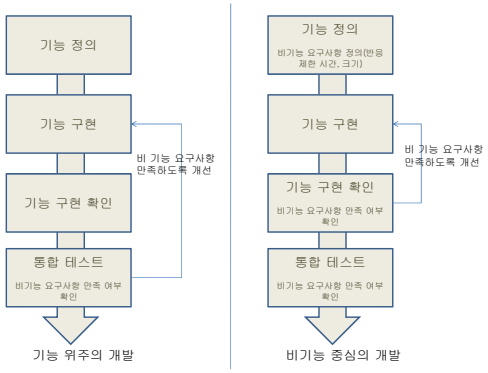
API(Application Program Interface)는 운영체계 및 응용 프로그램에서 사용하는 시스템에서 제공하는 기능들의 집합이다. 이러한 API가 제공하는 기능을 숙지한 후에 소프트웨어 개발자는 자신의 코드에서 API를 호출하는 방식으로 프로그램을 한다. 일반적으로 API에 대한 정보는 기능적인 명세만 제공된다. 그러므로 개발자는 실제 API를 사용할 때에 수행시간이 얼마나 걸릴지에 대한 구체적인 정보가 없어서 자신이 개발한 모듈의 수행 성능은 소프트웨어에 대한 코딩이 완료된 후에 직접 수행함으로써 테스트하는 방법밖에 없었다.

본 연구는 일반 업무용 소프트웨어 시스템의 개발에 있어서 비 기능적 요구사항으로 다루어지는 성능 관련 항목을 시스템 개발과정에 강제 시킴으로써 예측 가능한 소프트웨어 시스템을 구성할 수 있게 한다. 비즈니스 모형 작성과정에서 필요한 소프트웨어 시스템의 성능 요구조건을 수치적으로 명시하고 이를 설계 모형 작성과정에 반영시키고 소프트웨어를 구성하는 컴포넌트의 성능 항목에 대한 속성을 형식화 시켜서 설계 모형 단계에서 성능 시험을 할 수 있도록 한다. 본 연구는 그 동안 간과되었던 개별 구성 요소의 성능 부분을 수치적으로 강제함으로써 개발 비용을 예측할 수 있게 한다. 또한 하드웨어 중립적인 시간 개념을 사용함으로써 소프트웨어가 작동하는 하드웨어의 성능에 맞는 적절한 소프트웨어를 개발해 낼 수 있도록 한다.

2. 본론

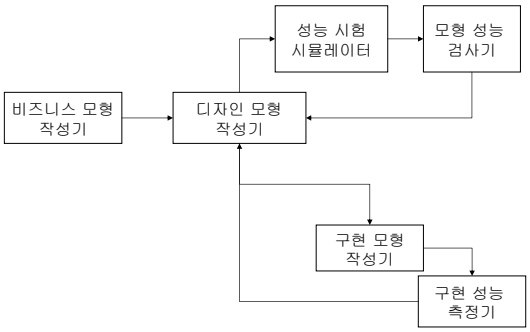
소프트웨어 개발 과정에서 일반적으로 모듈이 처리해야 할 기능을 정의하고 이 기능을 어떻게 만족시킬 것인가에 주안점을 두어서 개발한다. 보통 소프트웨어의 개발 절차는 요구 분석을 통해서 개발하여야 할 기능을 분석하고 이 기능을 묶어서 모듈로 정의한다. 모듈들간의 상호 인터페이스를 정의하고 각 모듈들은 코딩이 완료되면 컴파일러를 이용하여 컴파일 한 후에 원하는 기능이 제대로 수행되는지 확인하는 과정을 거친다. 이 과정에서 기능이 수행되지 않을 경우이거나 원하는 수행 속도가 나오지 않을 경우에는 프로그래밍 언어로 코딩 되어 있는 소스코드를 다시 수정한 후에 컴파일을 하여 다시 프로그램을 수행시켜 결과를 확인하는 과정을 반복하게 된다.

<그림 1>은 기존의 기능 중심의 개발 방법과 비기능 중심의 개발 방법을 비교한 그림이다.



<그림 1> 소프트웨어 개발 방식의 비교

이러한 소프트웨어 개발 과정을 지원하는 모형화 시스템의 구성도는 <그림 2>와 같은 요소로 구성된다.



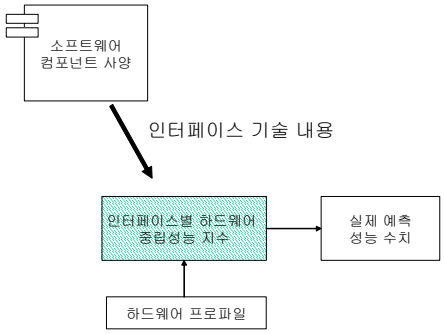
<그림 2> 비기능 중심 개발을 지원하는 소프트웨어 모형화 시스템의 구성

비즈니스 모형 작성기는 업무용 시스템을 필요로 하는 사람의 요구사항을 UML방식으로 표현한다. 이러한 요구사항 모델은 업무의 절차과정을 UML의 Use Case 다이어그램, 활동도, 순차도, 상태전이도, 객체다이어그램 등으로 표현한다. 이때 사용자의 요구사항 중에 비기능적 요구사항인 시간 성능 부분은 필요한 경우에 순차도 및 상태전이도, 활동도 등의 관계에 표현된다.

이와 같이 표현된 모델을 바탕으로 디자인 모형 작성기에서는 설계 모형을 작성하게 된다. 설계 모형은 비즈니스 모형의 요구사항을 만족시키는 소프트웨어 시스템의 모듈을 설계한다. 설계한 소프트웨어 시스템은 모형 성능 시험기를 통해서 비즈니스 모형에서 기술된 성능 조건을 만족시키는지 시험한다. 시험한 결과 성능조건을 만족시키지 못할 경우 구현 모델을 생성시키지 못하게 한다. 일단 설계 모형에서 성능조건을 만족시키게 되면 이 모형을 이용하여 구현 모형을 자동으로 생성한다. 자동으로 생성된 구현 모형은 소프트웨어를 실행시키기 위한 소스 코드를 포함하고 있다. 소스 코드를 하드웨어에서 실행시키기 위한 실행 코드로 컴파일 하고 난 후에 실행을 시켜서 지정된 성능대로 동작이 이루어지는지 확인한다. 만약 설계 모형에서 지정한 성능보다 떨어질 경우에는 디자인 모형 작성기로 강제로 재 작성하게 강제한다.

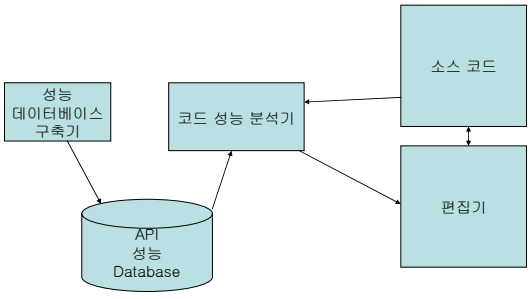
이러한 과정을 통해서 비즈니스 모형에서 명시한 비기능적 요구조건을 만족시키는 개별 소프트웨어 컴포넌트들이 개발된다.

<그림 3>은 이러한 절차에 사용되는 소프트웨어 컴포넌트의 성능 사양을 명세 하는 방법이다. 컴포넌트는 인터페이스들로 이루어져 있으며, 개별 인터페이스 별로 최대 수행 시간과 최소 수행 시간이 명시된다. 시간의 수치는 하드웨어 중립적인 수치로 표기되며, 이는 소프트웨어 컴포넌트가 수행될 개별 하드웨어 플랫폼의 프로파일을 토대로 재 계산되어 실제 수행시의 예측 성능을 산출한다.



<그림 3> 시간 성능 명세가 포함되는 컴포넌트의 명세

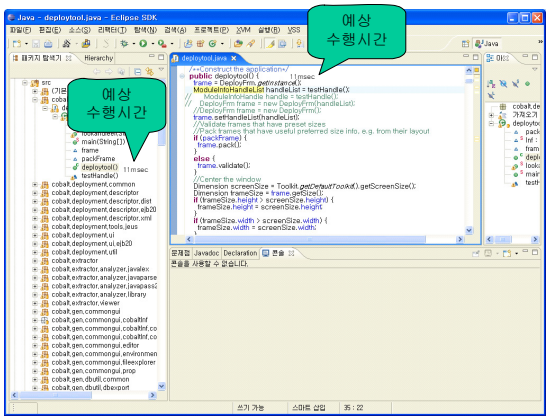
본 연구에서는 비 기능적 요구사항 중에서 성능 관련 항목을 중심으로 시스템 개발과정에 코드 작성과정 중에 성능 항목을 확인할 수 있도록 하는데 초점을 맞추었다. 구현 코드가 사용하는 API들의 성능 정보와 코드 자체에 대한 성능 분석정보를 이용하여 해당 모듈의 성능분석을 하고 이를 개발 과정중의 편집기에 디스플레이 함으로써 개발자로 하여금 작성 중의 코드의 성능을 실제 수행하지 않고서도 확인할 수 있게 함으로써 요구되는 성능에 맞는 코드를 작성할 수 있게 한다. <그림 4>는 구현 단계에서 성능 조건을 강제하는 소프트웨어 모형화 시스템의 구성도이다.



<그림 4> 성능조건을 강제하는 소프트웨어 개발 시스템의 구성도

성능 데이터베이스 구축기는 사용하려는 라이브러리의 API의 테스트 케이스를 수행시켜 WCET(Worst Case Execution Time)을 측정하여 데이터베이스에 저장한다. 소프트웨어는 수행되는 CPU의 성능에 따라서 실제 수행시간이 달라지므로 수행시간/CPU성능 과 같은 중립적인 인덱스를 만들어서 이를 이용한 인덱스값만 저장하도록 한다. 해당 성능 인덱스는 성능 데이터베이스에 기록하도록 한다. API 성능 데이터베이스는 라이브러리 및 운영체계가 제공하는 모든 API에 대한 성능 정보가 담겨져 있다.

코드 성능 분석기는 모듈의 소스코드를 입력받아서 API 성능 데이터베이스의 성능 자료를 이용하여 분석하려는 모듈의 성능을 계산한다. 편집기는 소스코드를 편집하고 코드 분석기를 이용하여 계산해낸 모듈의 성능을 개발자에게 제공한다. 개발자는 <그림 5>와 같은 편집기를 이용하여 현재 작성하고 있는 모듈의 성능을 해당 모듈을 컴파일 하여 실행하지 않고서도 예측할 수 있다.



<그림 5> 성능조건을 개발 중에 확인하는 소프트웨어 개발 편집기

3. 결론

본 연구는 대규모의 일반 업무용 소프트웨어를 개발하는데 있어서 그동안 간과되었던 개별 구성요소의 성능 부분을 수치적으로 강제함으로써 개발 비용을 예측할 수 있게 한다. 또한 하드웨어 중립적인 시간 개념을 사용함으로써 소프트웨어가 작동하는 하드웨어의 성능에 맞는 적절한 소프트웨어를 개발해 낼 수 있도록 한다. 이러한 기능은 본 발명이 제시하는 모형화 도구의 모형 성능 시험 및 성능 강제기의 작용에 의해서 개발 과정에 적용된다.

설계 단계에서 이러한 컴포넌트의 성능 시간 요소를 명세하고 개별 컴포넌트를 성능 시간 요소를 만족시킬 수 있도록 확인하여 개발하여 나가도록 강제 시킬 수 있다. 이러한 모형화 과정을 지원할 수 있도록 각 모형 작성기에서 성능 요소를 정형적으로 반드시 기술하게 하고 기술된 경우에만 설계를 할 수 있도록 강제함으로써 시스템 전체의 성능을 예측 가능하게 개발할 수 있다.

이로 인하여 소프트웨어를 개발하는데 있어서 개발자로 하여금 개발되는 소프트웨어의 코드를 작성하는 도중 즉시 모듈의 성능을 손쉽게 예측할 수 있도록 함으로써 소프트웨어의 성능을 유지시킬 수 있도록 한다

본 연구에서는 예측 가능한 시스템을 개발이 가능하도록 개별 컴포넌트를 표현하는 속성 항목에 성능 시간 관련 항목을 추가한다. 이와 같이 품질 지표가 들어간 컴포넌트를 이용하여 소프트웨어를 개발함으로써 설계 단계에서 소프트웨어의 성능을 시뮬레이션을 가능케 한다.

[참고문헌]

[1] Dawid Weiss and Marein Zduniak, Automated Integration Tests for mobile Application in Java 2 Micro Edition, LNCS, Springer Belin, pp478-487, 2007.

[2] 소프트웨어 테스트 전문기술, TTA, 2003.

[3] Atif M. Memon, A Comprehensive Framework for Testing Graphical User Interfaces, Ph.D. Dissertation, Univ. of Pittsburgh, 2001.